Optimization Internship

Supervisor: Beniamin Bogosel - Student: Mehdi Makni

July - August 2021

Introduction

The problem deals with the Optimal Partitioning of subsets (ω_i) in a domain Ω taking into account that the cost function would be the sum of the first eigenvalues of the Dirichlet-Laplace operator of these cells. More formally,
Given a set $\Omega \subseteq \mathbb{R}^n$ (the cases studied are $n = 1$ and $n = 2$) and an integer $N \in \mathbb{N}$, we want to subdivide Ω into N cells (ω_i) (Note that this implies $\forall i \neq j, \omega_i \cap \omega_j = \emptyset$ such that:

$$\min_{\omega_i \subseteq \Omega} \sum_{i=1}^{N} \lambda_1(\omega_i)$$

where $\lambda_1(\omega_i)$ corresponds to the first eigenvalue when solving the following partial differential equation:

$$\begin{cases} \Delta u = -\lambda(\omega_i) u \\ u = 0 \text{ on } \partial\omega_i \text{ that is } u \text{ cancels on the boundaries of the set.} \end{cases}$$

Motivation: The study of Optimal Partitioning based on spectral information is motivated by many applications. This ranges from biological models of competing species as described in NEHARI'S PROBLEM AND COMPETING SPECIES SYSTEMS to stable configuration for chemical reaction as in Minimization of the Renyi entropy production.

Part1: Theoretical Results

1 Dimension

First we start with the equivalent problem when we work in dimension 1:
The problem is equivalent to solving the differential equation on $[0, L]$:

$$\begin{cases} u'' = -\lambda u \\ u(x) = 0 \text{ for } x = 0 \text{ or } x = L \end{cases}$$

Since we are solving for the first eigenvalue, we have that $\lambda > 0$.
Hence the solution is of the form:

$$u(x) = A\cos(\sqrt{\lambda}x) + B\sin(\sqrt{\lambda}x)$$

Since $u(0) = 0$. This gives

$$u(x) = B\sin(\sqrt{\lambda}x)$$

Since $u(L) = 0$. This gives

$$\lambda = k^2\frac{\pi^2}{L^2}, k \in \mathbb{N}$$

Therefore:

$$\lambda_1 = \frac{\pi^2}{L^2}$$

Therefore if we are given N cells to subdivide an interval of length L we are trying to find $l_1, l_2, \ldots, l_N$ such that $\sum_{i=0}^{N} l_i = L$ and

$$\min_{l_1,\ldots,l_N} = \sum_{i=0}^{N} \frac{\pi^2}{l_i^2}$$

which is equivalent to minimizing

$$\min_{l_1,\ldots,l_N} = \sum_{i=0}^{N} \frac{1}{l_i^2}$$

Thanks to Cauchy-Schwartz, we have:

$$\sum_{i=0}^{N}\frac{1}{l_i^2}\cdot\sum_{i=0}^{N}1 \geq \left(\sum_{i=0}^{N}\frac{1}{l_i}\right)^2 = \frac{1}{L^2}\left(\sum_{i=0}^{N}\frac{1}{l_i}\right)^2\left(\sum_{i=0}^{N}l_i\right)^2 \geq \frac{N^4}{L^2}$$

The equality is attained if and only if $\quad \forall i, \quad l_i = \frac{L}{N}$.
That is the solution for 1D is to subdivide equally the initial interval.

2 Dimensions

In two dimensions the differential equation is given by:

$$\begin{cases} \Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\lambda(\omega_i)u \\ u = 0 \text{ on } \partial\omega_i \end{cases}$$

That is u cancels on the boundaries of each cell.
First we begin by stating a few remarks about the regions that will help our understanding.

- The first eigenvalue is a decreasing function of the shape. That is if $\omega \subseteq \Omega$ then $\lambda_1(\omega) \leq \lambda_1(\Omega)$. Indeed given a bigger subset, we can set the function to 0 everywhere in $\Omega \setminus \omega$. The function is not identically zero in ω and hence we get that the eigenvalue in Ω is at least equal to that of ω. this comes from the characterization of the eigenvalue as the minimum of the Rayleigh quotient.

- The regions of a cell must be continuous. Indeed if $\omega = \omega_1 \cup \omega_2$ such that $\omega_1 \cap \omega_2 = \emptyset$. That is they do not intersect at all including on boundaries, then $\lambda_1(\omega) = \max(\lambda_1(\omega_1), \lambda_1(\omega_2))$. Without loss of generality, $\lambda_1(\omega_1) \geq \lambda_1(\omega_2)$. In which case, it makes sense to let $\omega = \omega_1$ and append ω_2 to another cell adjacent to it and decreasing the first eigenvalue of that set and thus decrease the total sum of the eigenvalues.

There is no known theoretical solution for optimal partitioning based on spectral information. Hence we take advantage of numerics to this end.

For further theoretical results, this summer school course can be consulted.

Part2: Numerics for 2 dimensions

Theoretical Algorithm Description

The first approach to the problem is to consider an $n \times n$ grid on which we are going to work. A given shape ω is then going to be discretized and represented by certain points in our grid.
In order to compute the cost function itself, we should be able to compute the first eigenvalue of the Dirichlet-Laplace operator for a given shape ω. To this end, we follow this idea:

Compute first eigenvalue for a given shape ω

- Since we are working with finite differences method, we construct the 2D Laplacian matrix A for the $n^2 \times n^2$ grid. It has the following shape:

$$A = \begin{pmatrix} -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 1 & -4 & 1 & 0 & \dots & 0 & 1 & \dots & \vdots \\ 0 & 1 & -4 & \ddots & \ddots & \dots & \dots & \ddots & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \dots & \dots & 1 \\ 0 & \dots & \ddots & \ddots & \ddots & \ddots & \ddots & \dots & 0 \\ 1 & 0 & \dots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 1 & \dots & \dots & \ddots & 1 & -4 & 1 & 0 \\ \vdots & \ddots & \ddots & 0 & \dots & 0 & 1 & -4 & 1 \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & -4 \end{pmatrix}$$

- Create a diagonal penalization matrix P of size $n^2 \times n^2$ where the i^{th} cell of the diagonal is equal to a penalization term $c >> 1$ if and only if that point in the grid does not belong to ω. Note that since we work $n \times n$ grid, there are n^2 points and they are numbered in sequential order that is a point at position (i, j) will have number $i + n\dot{j}$ (or the other way around depending on implementation). That's what explains the dimension of P as well as the dimension of the Laplacian matrix A.

- Solving the problem $(A + P)v = \lambda v$ to find the first eigenvalue.
- If c is large enough we get an approximation for the Dirichlet boundary condition.

Input Description

Theoretically the implemented algorithm requires as input:

- n the dimension of the grid
- c the penalization term
- N the number of cells

Each cell i where $i \in \{1, \dots, n\}$ is going to represent a density function. That is to say it is an n^2-dimensional vector that represents the likelihood that every point in the grid belongs to that cell i. For the optimization we run gradient descent to optimize the cost function: sum of first eigenvalues of Dirichlet-Laplace operator.
If we therefore denote by:

$$X = \begin{pmatrix} \Phi_1 & | & \Phi_2 & | & \dots & | & \Phi_N \end{pmatrix}$$

which has shape $n^2 \times N$ we have that any row of X adds up to 1. That translates the fact that for a given point the sum of probabilities of belonging to one of the regions adds up to 1.
In the algorithm X is initialized randomly with uniform values in $[0, 1]$ such that the above-mentioned property is preserved.

Algorithm 1 Optimization Loop:

procedure GIVEN X
  for $i = 1 : N$ **do**:
    select column i which represents density Φ_i.
    solve $[-\Delta + c(1 - \Phi_i)]u = \lambda_1 u$ (see previous subsection for details)
    $A \leftarrow L + diag(c \cdot (1 - \Phi_i))$
    obtain first eigenvalue of A denoted by $\lambda_1(c, \Phi_i)$
    obtain associated eigenfunction u_i
    $\nabla\lambda_1(c, \Phi_i) = -cu_i^2$
    $\Phi_i \leftarrow \Phi_i + \alpha cu_i^2$
  $\Phi_i \leftarrow \frac{\Phi_i}{\sum_{j\in\{1,N\}} \Phi_i}$

The proof of the gradient formula is given here.

Numerical Techniques for efficiency improvement

In order to have a code that is accessible to a wide audience, it was implemented in python taking advantage of the well known libraries:

- numpy and scipy for scientific computations

- matplotlib for plotting. It displays the evolution of the cells and give the final optimal partitioning

- For the Laplacian matrix L as well as the penalization matrix P, we use the scipy sparse matrix format in order to reduce the memory consumption. In particular we use diags (link).
- For the computation of the first eigenvalue we use eigsh(link). This is possible because all our matrices are symmetric.
These details are of paramount importance if we want to run the algorithm with a high resolution.
- Another optimization technique we used is Gradient Descent with variable step. It makes the algorithm converge faster.

Advanced Improvements for the algorithm

Coarse Grid

The basic idea is to start iterating on a low resolution grid, that is small n, and then interpolate the result to a higher resolution grid, that is big n, and continue the iteration with a very good initialization. This is because a higher resolution requires much more time to evolve. The interpolation is used with interp2d(link)
Another idea with the same pattern is to start with a smaller penalization term and increase as later stages. A Bigger penalization term gives stricter boundaries but takes more time to run.

Neighbor-Only penalization

The idea is to locate the neighbors of a given shape ω and penalize them only instead of the whole grid other than ω. This has the advantages of:
- better conditioning for the matrix
- smaller matrix for which we compute the first eigenvalue

Finding the neighbors In order to find the neighbors of a shape ω we follow this idea:
- Consider the grid as a graph where edges are between neighboring points. That is a point in the middle of the grid should have 4 neighbours. The adjacency matrix of this graph is similar to the Laplacian but with zeros on the diagonal entries. We denote it by

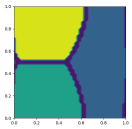
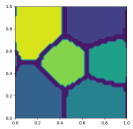
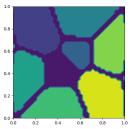
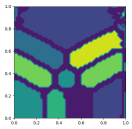
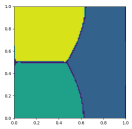
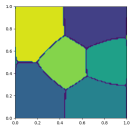
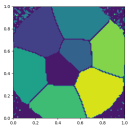
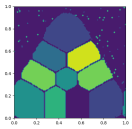
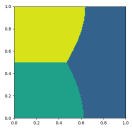
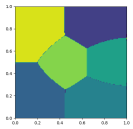
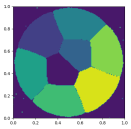
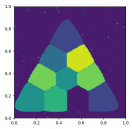
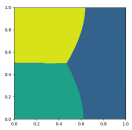
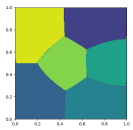
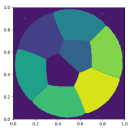
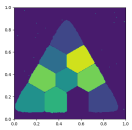
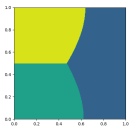
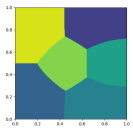
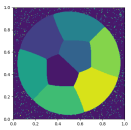
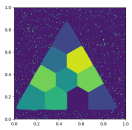
$$N = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & \dots & 0 & 1 & \dots & \vdots \\ 0 & 1 & 0 & \ddots & \ddots & \dots & \dots & \ddots & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \dots & \dots & 1 \\ 0 & \dots & \ddots & \ddots & \ddots & \ddots & \ddots & \dots & 0 \\ 1 & 0 & \dots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 1 & \dots & \dots & \ddots & 1 & 0 & 1 & 0 \\ \vdots & \ddots & \ddots & 0 & \dots & 0 & 1 & 0 & 1 \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & 0 \end{pmatrix}$$

- Take this matrix to the power k (taken to be 5). This way any entry in row i that

is positive represents a point in the graph that we can reach with k steps from point i. Hence it is considered a neighboring point.
- We multiply this matrix N^k with shape $n^2 \times n^2$ with the vector of booleans with True values where a point belongs to the shape ω. This vector has length n^2.
- The index of any value that is positive in the resulting vector represents a point in the neighbours of ω.
- Now we compute the first eigenvalue and the corresponding eigenvector with the smaller matrix with shape $n_i \times n_i$ where n_i is the length of the points of the shape ω as well as the union of their neighbors up to k steps.

Sample Results from the Algorithm

Table 1: Outputs of the Algorithm at different stages of the Optimization

Resolution / Penalization	Square (3)	Square (6)	Disk (8)	Triangle (10)
50 / 10^3				
100 / 10^4				
150 / 10^5				
250 / 10^5				
300 / 10^6				

References

[1] Richard S. Laugsesn. 2016 CRM Summer School Minicourse: Spectral Theory of Partial Differential Equations.

[2] M. Conti, S. Terracini, and G. Verzini. NEHARI'S PROBLEM AND COMPETING SPECIES SYSTEMS

[3] O. Cybulski, V. Babin, and R. Hoł yst. Minimization of the Renyi entropy production in the space-partitioning process.

[4] Beniamin Bogosel. Efficient algorithm for optimizing spectral partitions.

[5] Beniamin Bogosel. Discrete version of an optimal partitioning problem.

[6] Blaise Bourdin, Dorin Bucur, and Edouard Oudet. Optimal partitions for eigenvalues